

# Verificação de Modelos (Model Checking)

Estes slides são baseados nas notas de aula da Profa. Corina  
Cîrstea

# Agenda

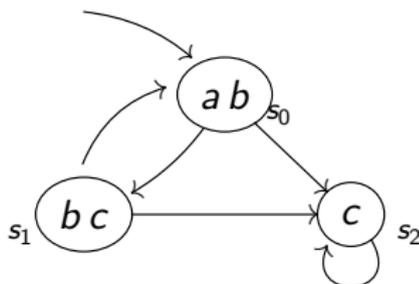
Lógica Temporal

Lógica de Árvore de Computação (CTL)

Verificação de Modelo do CTL

# Caminhos de Computação versus Árvores de Computação

- ▶ Estrutura Kripke:



- ▶ suposição adicional: cada estado tem no mínimo um sucessor  
⇒ processos infinitos !

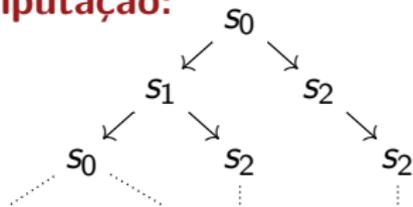
- ▶ alguns **caminhos de computação**:

$s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \rightarrow \dots$

$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow \dots$

$s_0 \rightarrow s_2 \rightarrow s_2 \rightarrow \dots$

- ▶ **árvore de computação**:



# Lógicas de Tempo Linear versus Tempo de Ramificação (Branching Time)

- ▶ modelos de tempo:

1. tempo linear: conjunto de todos os caminhos de computação
2. tempo de ramificação: árvore de computação

- ▶ lógicas temporais:

1. lógicas de tempo linear:

- ▶ fórmulas avaliadas em caminhos de computação (estrutura de ramificação é abstraída)
- ▶ e.g. “para todos os caminhos,  $p$  detém em algum ponto ao longo do caminho”

2. lógicas de tempo de ramificação:

- ▶ fórmulas avaliadas em árvores de computação
- ▶ e.g. “sempre que  $p$  ocorre, é possível alcançar um estado onde  $q$  ocorre”

## Exclusão Mútua: Propriedades de Corretude Revisadas

- ▶ **exclusão mútua**: no máximo um processo na região crítica a qualquer momento (i.e. *em todos os estados alcançáveis*)
- ▶ **starvation freedom**: *sempre que* um processo tenta entrar em sua região crítica, ele irá *eventualmente* suceder (*ao longo de cada caminho de computação*)
- ▶ **sem bloqueio** (vivacidade): um processo pode sempre, em algum ponto no futuro (*ao longo de algum caminho de computação*), solicitar para entrar em sua região crítica

As *propriedades temporais* acima devem se manter *em todos os estados iniciais*.

# Recap em Lógica Proposicional

- ▶ assumo um conjunto *Prop* de proposições atômicas
- ▶ **sintaxe:**

$$\phi, \psi ::= p \mid \text{tt} \mid \neg\phi \mid \phi \wedge \psi \quad (p \in Prop)$$

Note: todos os outros operadores booleanos são definíveis, por exemplo

$$\text{ff} ::= \neg\text{tt}$$

$$\phi \vee \psi ::= \neg(\neg\phi \wedge \neg\psi)$$

$$\phi \rightarrow \psi ::= \neg(\phi \wedge \neg\psi)$$

- ▶ **modelos** são *avaliações*, dizendo quais proposições são verdadeiras e quais não são:

$$V : Prop \rightarrow \{\text{True}, \text{False}\}$$

- ▶ **significado** (*semântica*) das fórmulas:
  - ▶ *V* define o significado das proposições atômicas
  - ▶ a tabela da verdade define o significado dos operadores booleanos

# Lógica de Árvore de Computação (CTL)

- ▶ sintaxe:

$$\phi, \psi ::= p \mid \text{tt} \mid \neg\phi \mid \phi \wedge \psi \mid \mathbf{AX} \phi \mid \mathbf{EX} \phi \mid \mathbf{AF} \phi \mid \mathbf{EF} \phi \mid \\ \mathbf{AG} \phi \mid \mathbf{EG} \phi \mid \mathbf{A} [\phi \mathbf{U} \psi] \mid \mathbf{E} [\phi \mathbf{U} \psi] \quad (p \in \text{Prop})$$

- ▶ leituras de fórmulas modais:

**AX**  $\phi$        $\phi$  holds in **All neXt** states

**EX**  $\phi$       there **Exists** a ne**Xt** state in which  $\phi$  holds

**AF**  $\phi$       along **All** paths,  $\phi$  holds in some **F**uture state

**EF**  $\phi$       there **Exists** a path along which  $\phi$  holds in some **F**uture state

**AG**  $\phi$       along **All** paths,  $\phi$  holds **G**lobally

**EG**  $\phi$       there **Exists** a path along which  $\phi$  holds **G**lobally

**A** [ $\phi$  **U**  $\psi$ ]      along **All** paths,  $\phi$  holds **U**ntil  $\psi$  holds

**E** [ $\phi$  **U**  $\psi$ ]      there **Exists** a path along which  $\phi$  holds **U**ntil  $\psi$  holds

- ▶ exemplos de fórmulas: **AG EF**  $\phi$ , **E** [**tt U AG**  $\phi$ ]

# Algumas Observações na Sintaxe da CTL

- ▶ os pares **AX** , **EX** , ... , **A[U]** , **E[U]** são indivisíveis !
- ▶ prioridades de ligação:
  - ▶  $\neg$  , **AG** , **EG** , **AF** , **EF** , **AX** e **EX** ligam mais firmemente
  - ▶  $\wedge$  e  $\vee$  estão próximos
  - ▶  $\rightarrow$  , **A[U]** , **E[U]** ligam menos firmemente
- ▶ e.g. **EF EG p**  $\rightarrow$  **AF r** significa ( **EF EG p** )  $\rightarrow$  **AF r**
- ▶ parênteses podem ser usados para substituir as prioridades acima, e.g. **EF ( EG p**  $\rightarrow$  **AF r** ) , **EF EG ( p**  $\rightarrow$  **AF r** )

# Semântica da CTL

Assuma a estrutura Kripke  $M = (S, R, V)$ .

A **semântica da CTL** define quando uma fórmula  $\phi$  da CTL é verdadeira *em um estado*  $s \in S$ .

Quando escrevemos  $s \models \phi$  (e dizemos que  $s$  **satisfaz**  $\phi$ ) se  $\phi$  é verdadeiro em  $s$ .

A definição de quando  $s \models \phi$  é por *indução na estrutura de  $\phi$* :

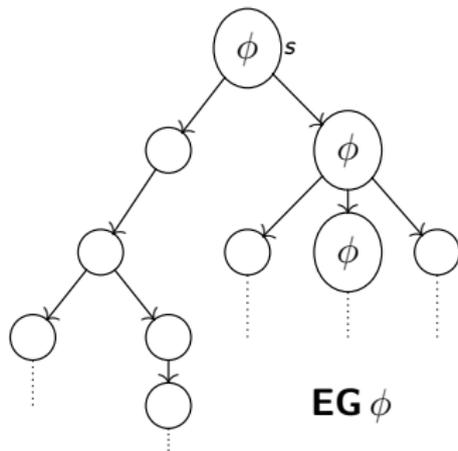
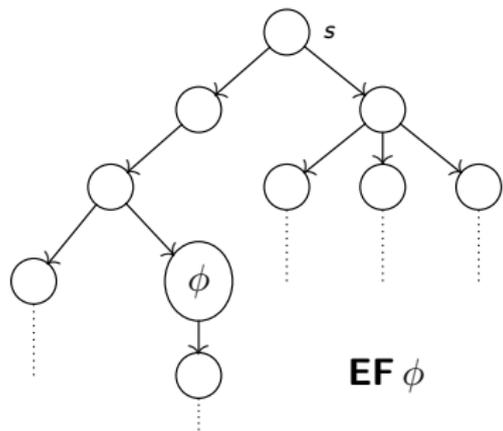
$s \models p$       sse     $V(p, s) = true$

$s \models \neg\phi$     sse     $s \models \phi$  não for verdadeiro

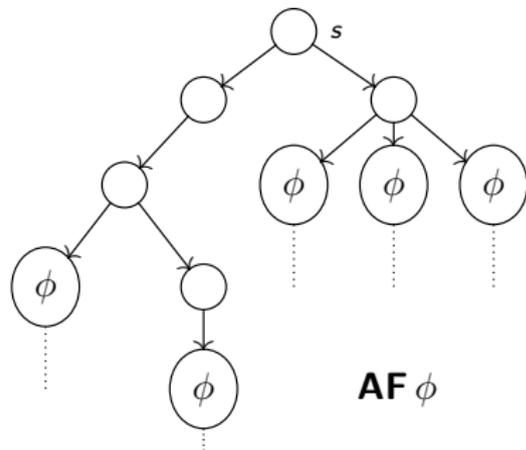
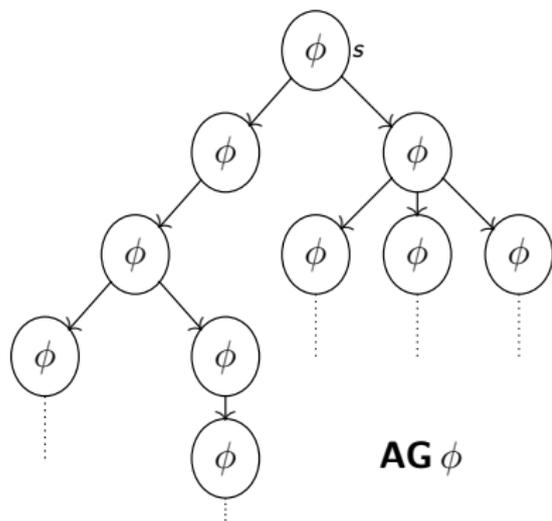
$s \models \phi \wedge \psi$     sse     $s \models \phi$  e  $s \models \psi$

...

# Semântica da CTL: Exemplo

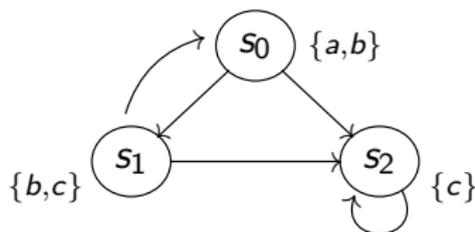


# Semântica da CTL: Exemplo





## Outro Exemplo



$$s_0 \models \mathbf{EX} (b \wedge c)$$

$$s_0 \models \neg \mathbf{AX} (b \wedge c)$$

$$s_0 \not\models \mathbf{EG} c$$

$$s_0 \models \mathbf{AG AF} (b \vee c)$$

$$s_0 \models \mathbf{A} [a \mathbf{U} c]$$

$$s_0 \models \mathbf{E} [(a \wedge b) \mathbf{U} c]$$

$$s_0 \not\models \mathbf{AG} (b \rightarrow \mathbf{EG} c)$$

**Note:**  $s \models \mathbf{AG AF} \phi$  sse  $\phi$  é verdadeiro infinitas vezes ao longo de cada caminho de  $s$ .

# Exclusão Mútua: um Modelo

```
int turn;
```

```
P = m : cobegin P0 || P1 coend m'
```

```
P0 = n0 : while True do
```

```
  t0 : wait (turn = 0);
```

```
  c0 : use resource; turn := 1;
```

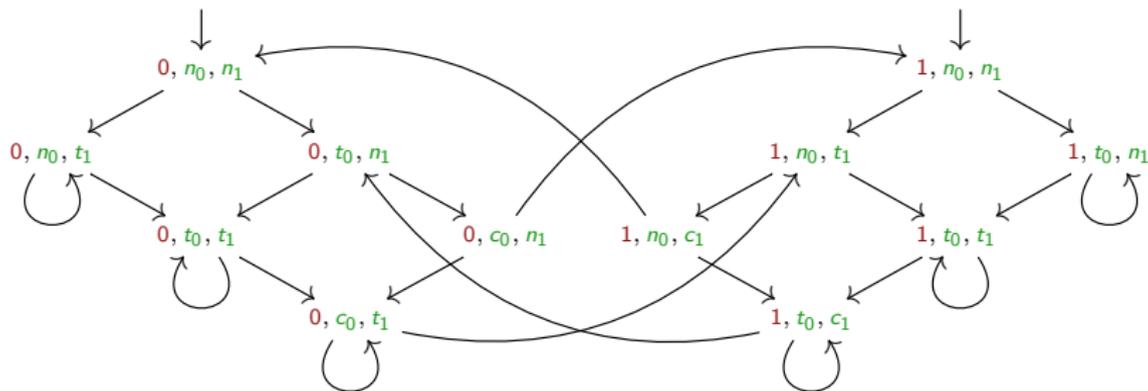
```
endwhile ; n'0
```

```
P1 = n1 : while True do
```

```
  t1 : wait (turn = 1);
```

```
  c1 : use resource; turn := 0;
```

```
endwhile ; n'1
```



# Exclusão Mútua: Propriedades de Corretude

Proposições atômicas:

$c_0, c_1$  (estado crítico)

$n_0, n_1$  (estado não crítico)

$t_0, t_1$  (tentando entrar no estado crítico)

- ▶ **exclusão mútua**: no máximo um processo na região crítica a *qualquer momento*

$$\mathbf{AG} \neg (c_0 \wedge c_1)$$

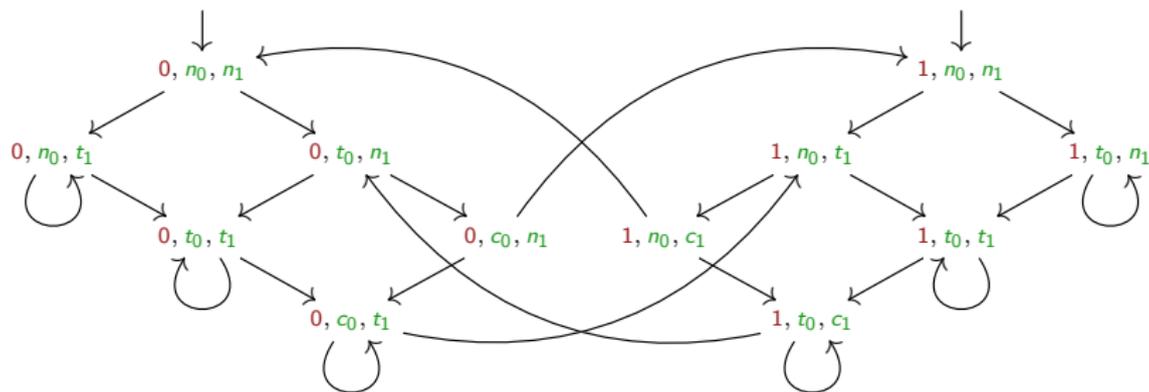
- ▶ **starvation freedom**: *sempre que* um processo tenta entrar na sua região crítica, ele *eventualmente* entrará

$$\mathbf{AG} ( (t_0 \rightarrow \mathbf{AF} c_0) \wedge (t_1 \rightarrow \mathbf{AF} c_1) )$$

- ▶ **sem bloqueio**: um processo pode sempre, em algum ponto no futuro, pedir para entrar em sua região crítica

$$\mathbf{AG} ( \mathbf{EF} t_0 \wedge \mathbf{EF} t_1 )$$

## Exclusão Mútua: Checagem da Corretude



**AG**  $\neg(c_0 \wedge c_1)$  ✓

**AG**  $((t_0 \rightarrow \mathbf{AF} c_0) \wedge (t_1 \rightarrow \mathbf{AF} c_1))$  ✗ ✓

(precisa do conceito de **weak fairness** para assegurar a propriedade!)

**AG**  $(\mathbf{EF} t_0 \wedge \mathbf{EF} t_1)$  ✓

**EF**  $(c_0 \wedge \mathbf{E}[c_0 \mathbf{U} (\neg c_0 \wedge \mathbf{E}[\neg c_1 \mathbf{U} c_0])]) \vee \dots$  ✗

(difícil verificar a mão!)

## Exercício

Assuma as seguintes proposições atômicas: *start*, *ready*, *requested*, *acknowledged*, *enabled*, *running*, *deadlock*.

Especifique as seguintes propriedades de corretude em CTL:

1. É possível/impossível de alcançar um estado onde *start* é verdadeiro mas *ready* não é.
2. Sempre que ocorre uma solicitação (*request*), ela irá eventualmente ser reconhecida (*acknowledged*).
3. Aconteça o que acontecer, *deadlock* irá eventualmente ocorrer.
4. A partir de qualquer estado, é possível alcançar um estado *ready*.

# Solução do Exercício

Fórmulas atômicas:

*start, ready, requested, acknowledged, enabled, running, deadlock.*

1. É (im)possível de alcançar um estado onde *start* é verdadeiro mas *ready* não é.

$$\mathbf{AG} \neg(\mathit{start} \wedge \neg \mathit{ready}) \qquad \mathbf{EF} (\mathit{start} \wedge \neg \mathit{ready})$$

2. Sempre que ocorre uma solicitação (*request*), ela irá eventualmente ser reconhecida (*acknowledged*).

$$\mathbf{AG} (\mathit{requested} \rightarrow \mathbf{AF} \mathit{acknowledged})$$

3. Aconteça o que acontecer, *deadlock* eventualmente ocorrerá.

$$\mathbf{AF} \mathit{deadlock}$$

4. A partir de qualquer estado, é possível alcançar um estado *ready*.

$$\mathbf{AG} (\mathbf{EF} \mathit{ready})$$

# Algumas Equivalências entre Fórmulas CTL

- ▶ **equivalência semântica:**  $\phi \equiv \psi$  se qualquer estado de qualquer modelo que satisfaz  $\phi$  também satisfaz  $\psi$ , e reciprocamente.
- ▶ algumas equivalências:

$$\mathbf{AX} \phi \equiv \neg \mathbf{EX} \neg \phi$$

$$\mathbf{AG} \phi \equiv \neg \mathbf{EF} \neg \phi$$

$$\mathbf{AF} \phi \equiv \mathbf{A} [\text{tt U } \phi]$$

$$\mathbf{EG} \phi \equiv \neg \mathbf{AF} \neg \phi$$

$$\mathbf{EF} \phi \equiv \mathbf{E} [\text{tt U } \phi]$$

$\implies$   $\mathbf{A} [- \mathbf{U} -]$ ,  $\mathbf{E} [- \mathbf{U} -]$ ,  $\mathbf{EX}$  suficiente para expressar todos os outros operadores CTL (**conjunto adequado de conectivos CTL**).

# Sub-conjunto Suficiente de Operadores CTL

**Teorema.** Um conjunto de conectivos temporais em CTL é adequado, se e somente se ele contiver:

- ▶ pelo menos um dos **AX**, **EX**
- ▶ pelo menos um dos **EG**, **AF**, **A[-U-]**
- ▶ **E[-U-]**

**Corolário.** Os operadores **EX**, **AF** e **E[-U-]** são suficientes para expressar todas os outros.

# Algoritmo de Verificação de Modelo para CTL

**Entradas:** estrutura Kripke finita  $M$ , fórmula CTL  $\phi$  contendo apenas **EX**, **AF** e **E[-U-]** como operadores temporais.

**Saídas:** os estados de  $M$  que satisfazem  $\phi$

## Algoritmo:

- ▶ Gerar todas as subfórmulas de  $\phi$  e ordená-las por complexidade (proposições atômicas primeiro,  $\phi$  por último).
- ▶ Repetir: pegar uma sub-fórmula da lista e *anotar* os estados de  $M$  que sejam satisfeitos por esta fórmula.
- ▶ Quando nós alcançarmos o fim da lista, sabemos quais estados satisfazem  $\phi$ .

## Como Anotar?

- ▶ A avaliação de  $V$  nos diz como anotar os estados com proposições atômicas
- ▶ Todos os estados são anotados com  $\text{tt}$ .
- ▶ Quando uma sub-fórmula atual é  $\neg\phi$ , nós anotamos com isto os estados que não estão com  $\phi$  (note que  $\phi$  precede  $\neg\phi$  na lista de sub-fórmulas, assim nós já anotamos os estados com  $\phi$ ).
- ▶ Da mesma forma para  $\phi_1 \wedge \phi_2 \dots$

## Como Anotar? (Continuação)

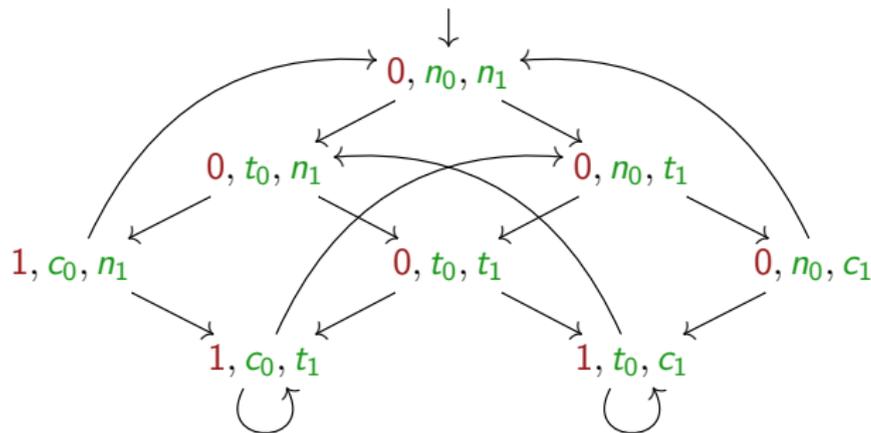
- ▶ Para **EX**  $\phi$ , anotar antecessores de qualquer estado rotulado  $\phi$  por **EX**  $\phi$ .
- ▶ Para **E** [ $\phi_1$  **U**  $\phi_2$ ], primeiro encontrar todos os estados anotados por  $\phi_2$  e anotá-los por **E** [ $\phi_1$  **U**  $\phi_2$ ]. Então trabalhe para trás a partir daqueles estados e todo tempo que nós encontrarmos os estados  $\phi_1$ , anotá-los com **E** [ $\phi_1$  **U**  $\phi_2$ ].
- ▶ Para **AF**  $\phi$ , primeiro anote todos os estados anotados com  $\phi$  por **AF**  $\phi$ . Então anote um estado com **AF**  $\phi$  se todos os seus sucessores estiverem anotados com **AF**  $\phi$ . Repita até que não exista nenhuma mudança.

### Notas:

- ▶ os operadores acima são suficientes para expressar todos os outros operadores do CTL,
- ▶ as equivalências semânticas dadas anteriormente precisam ser usadas quando a fórmula a ser verificada contém os operadores temporais que não **EX**, **AF** e **E** [**\_** **U** **\_**].

## Exercício

Considere o seguinte modelo de exclusão mútua:



Verifique se o estado inicial deste modelo satisfaz as fórmulas

$E[\neg c_1 \text{ U } c_0]$ ,  $AF c_0$  e  $EX t_0$ .

## Observações sobre o Algoritmo

- ▶ Existe uma variante do algoritmo que considera apenas os caminhos de computação que são justos.
- ▶ A complexidade de tempo do algoritmo é quadrático no tamanho do modelo (número de estados + número de transições) e linear no tamanho da fórmula (número de conectivos).
- ▶ Existe uma melhoria do algoritmo (substituindo **AF** por **EG**) cuja complexidade de tempo é linear no tamanho do modelo e no tamanho da fórmula.
- ▶ Isto ainda *não é bom o suficiente*, tendo em vista que o tamanho do modelo é **exponencial** no número de variáveis e no número de componentes paralelos . . .
- ▶ O verificador NuSMV usa estruturas de dados chamada de **OBDDs (ordered binary decision diagrams)** para representar *conjunto de estados* em vez de *estados individuais*.